An Ant System Approach to Markov Decision Processes

Hyeong Soo Chang[*], Walter J. Gutjahr[†], Jihoon Yang[‡], and Sungyong Park[‡]

September 8, 2003

Abstract

In this paper, we develop ant-system based algorithms for approximately solving large Markov Decision Process (MDP) problems, a.k.a. stochastic dynamic programming, for infinite horizon discounted cost criterion, extending the applicability of the ant-system metaheuristic into *dynamic stochastic* optimization problems. We consider both cases of known and unknown state transition dynamics of given MDP problems. The algorithms inherit the spirit of the well-known policy iteration algorithm with an adaptation of the ant system into MDP settings with some modifications and extensions, while preserving the probabilistic convergence property of the ant system.

Keywords: Ant system, Markov decision process, reinforcement learning, evolutionary algorithm, simulation

[*]Correspondence author. H. S. Chang is with Department of Computer Science and Engineering at Sogang University, Seoul, Korea and can be reached by e-mail at hschang@sogang.ac.kr, by phone +82-2-705-8925, or by fax +82-2-704-8273.

[†]W. J. Gutjahr is with Department of Statistics and Decision Support Systems, University of Vienna, Vienna, Austria and can be reached by email at walter.gutjahr@univie.ac.at.

[‡]J. Yang and S. Park are with the same affiliation with H. S. Chang and can be reached by email jhyang@ccs.sogang.ac.kr and parksy@sogang.ac.kr, respectively.

1 Introduction

A novel "general purpose" or *metaheuristic* algorithm based on the foraging behavior of ant colonies has been applied, with suitable modifications, to various *deterministic* combinatorial optimization problems to break *the curse of dimensionality* in solving them and it has been reported that the results are successful in many problems (see, e.g., [9] [10] for surveys). Recently, it has been shown in [15] that the solutions generated at each iteration of an ant-system based algorithm converge to the unique optimal solution of a given combinatorial optimization problem, under certain conditions, with a probability that can be made arbitrarily close to 1.

In this paper, we develop ant-system based algorithms for approximately solving Markov Decision Processes (MDPs) [20], a.k.a. stochastic dynamic programming, for infinite horizon discounted cost criterion based on the ant-system framework of [15], extending the applicability of the ant system metaheuristic into *dynamic stochastic* optimization problems. The model of MDPs is widely used for solving sequential decision making problems that arise in various areas such as telecommunication networks, financial engineering, ecology, statistical physics, manufacturing systems, operations research, bioinformatics, artificial intelligence, and etc. to just name a few (see, e.g., [1, 20, 2] for many interesting example problems).

A decision maker perceives the world or the environment s/he belongs to at each decision time where the world changes probabilistically. Based on his/her description of the world or information of the perception or *state*, s/he makes a decision to react to the environment and obtains a certain reward/cost associated with his/her decision and the state of the world. How the world changes for the next decision of the decision maker (state transition dynamics) depends on which decision has been made at which state by the decision maker now. The goal of the decision maker is obtaining a decision rule or *policy*, which is a mapping from states to available decisions, to maximize/minimize the expected (discounted) rewards/costs sum over stochastic changes of the world over infinite horizon. Unfortunately, a stochastic control optimization problem formulated by an MDP experiences the curse of dimensionality, making it very impractical to apply the well-known exact algorithms, e.g., Value Iteration (VI) and Policy Iteration (PI) [20], to solve the MDP. The time-complexity depends on the size of the state space and the size of available decisions or action space making the complexity blow up as either case increases exponentially. There are numerous approximation-based schemes to get away with the dimensionality problem via various techniques, e.g., structural analysis, aggregation, sampling, feature extraction, etc. see, e.g., [3, 20, 21]. However, to the best of the authors' knowledge, there is few algorithm that incorporates the success of "evolutionary computation" into solving MDPs except the recent works of Chang et al. [5] [4] based on Genetic Algorithm (GA) [14] and Simulated Annealing (SA) [16].

Ant System was developed by Dorigo and other authors (see [11]) and has been generalized later on to the Ant Colony Optimization (ACO) metaheuristic approach [9] which has found numerous applications in diverse fields in the meantime. It is a constructive and repetitive heuristic technique where the constructive part is similar to greedy heuristics, and the repetitive part draws from concepts in Reinforcement Learning [21]. The best way of explaining the basic ideas of Ant Colony Optimization is by reference to a biological metaphor which also gave the approach its name: the search behavior of biological ant colonies. In nature, ants move around in their environment in a rather random way, but they have a certain tendency to follow the walk of other ants. They can recognize these walks because, while moving, each ant leaves a chemical substance called *pheromone* on the ground. Sensing pheromone on a path increases the probability of an ant to follow it, which further reinforces this path. This mechanism has the effect that *short* paths between a starting point and a goal point are favored, leading to a kind of heuristic optimization behavior.

In ACO, the described principle is exploited for optimizing an arbitrary objective function of a combinatorial problem by simulating the walks of conceptual ants and by doing the reinforcement of good walks based on a post-hoc evaluation of the objective. In the ant system [15] that we consider in this paper, a directed graph (called "Construction Graph") is used as the basic structure on which the walks take place. The feasible solutions of the combinatorial optimization problem are encoded as paths on this graph. In each iteration, several ants are first positioned at the same node of the graph. Then, in successive steps, each ant chooses randomly a successor node of the node where it is currently located. The probabilities for the successor nodes are computed from "pheromone values" assigned to the arcs of the graph (and additionally from so-called "visibility values", which are pre-evaluations of the suitability of possible continuations). After all ants have finished their walks, the walks are decoded as problem solutions and evaluated. The arcs on "good" walks are then reinforced by pheromone increments. This process is iterated. We will express the procedure in more formal terms further below (see Section 3).

The algorithms proposed in this paper inherit the spirit of the PI algorithm for solving MDPs exactly. Loosely speaking, the basic algorithmic procedure is as follows: each ant generates its candidate policy (traversing a directed graph) and evaluates its policy ("policy evaluation" step in PI). Temporal Difference (TD) learning [21] [3] is used, if the state transition dynamics is unknown, to heuristically evaluate the quality of a given policy. We refer to the resulting algorithm as ANT-TD. If known, the usual policy evaluation step is followed and the resulting algorithm is referred to as ANT-PI. The (estimate) quality of each ant's candidate policy is merged to deposit a new pheromone, which in turn is used for each ant to generate a new candidate policy. During the merge step of the new pheromone generation, an *elite* policy is generated via a method called "policy switching" [6], where the elite policy is a policy that improves the performances of all of the candidate policies generated so far by ants, based on an idea similar to the "*elitist strategy*" concepts [11], and the quality of the elite policy is used for the new pheromone generation.

Contrary to PI, by directly manipulating the policies, both of the ANT-PI and the ANT-TD algorithms eliminate the operation of minimization over the entire action space in the "policy improvement" step of PI and a certain monotonicity property is preserved over the elite policies. Therefore, the proposed algorithms would be effective in the domains where the action state space is finite but extremely large. If a local search is used, it cannot guarantee that a global minimum has been found. The main contribution of the present paper is the use of a (random) evolutionary search algorithm in the context of MDPs with a probabilistic convergence guarantee. ANT-PI preserves a certain monotonicity property via elite policies and converges to the optimal policy with a probability that can be made arbitrarily close to 1.

This paper is organized as follows. We start with some necessary background on MDPs in Section 2. In Section 3, we formally describe ANT-PI with detailed discussion and in Section 4, we describe ANT-TD. In Section 5, we discuss some complexity issue of ANT-PI compared with PI. We then conclude the present paper with some remarks in Section 6.

2 Background

2.1 Markov Decision Processes

We briefly provide the formal model of Markov decision processes. For a substantial discussion, see, e.g., [20].

An MDP M is 4-tuple (X, A, P, C), where X is a finite set of states in the system (the information that is available to a decision maker for the world he perceives) and A is a set of actions or decisions. At state x, not all actions in A can be taken. We let $A(x)$ denote the finite set of *admissible* actions in A at state $x \in X$. P is a state transition function that describes how the system evolves over time such that it is a mapping from $X \times A(X)$ to a distribution over X. We denote the probability that a state $x \in X$ makes a transition to another state $y \in X$ by taking an action $a \in A(x)$ as $P(y|x, a)$. The probabilities of transitions to the next states depend only on the current state and action, but not the past history of the system, which makes the decision making processes be Markovian. C is a bounded cost function that maps $X \times A(X)$ to a real number in $\mathcal{R}$.

We define a policy or decision rule π as a mapping from X to $A(X)$ and let Π be the set of all possible policies. That is, a policy provides what action needs to be taken at which state to the decision maker at each discrete decision time.

Let $X_t, t = 0, 1, 2, \cdots$ be a random variable that denotes the state at time t. For a given policy $\pi \in \Pi$, starting with a given initial state $X_0 = x$, the decision maker follows the policy π over time such that a particular system path that the decision maker follows is given as a sequence of state and action $(X_0 = x_0, a_0, X_1 = x_1, a_1, \cdots, X_t = x_t, a_t, X_{t+1} = x_{t+1}, a_{t+1}, \cdots)$, where a_t is the action taken at time t and $a_t = \pi(x_t)$ with $x_t \in X$. Over the system path, the system accumulates the discounted costs of

$$\sum_{t=0}^{\infty} \gamma^t C(x_t, \pi(x_t)), 0 < \gamma < 1,$$

and the probability that the decision maker follows the particular path over time can be obtained as $\prod_{t=0}^{\infty} P(x_{t+1}|x_t, \pi(x_t))$. Therefore, we can take the expectation of the discounted costs over all possible system paths with those probabilities. Formally, we define *the value of following a policy* $\pi \in \Pi$ as

$$V^{\pi}(x) = E\left[\sum_{t=0}^{\infty} \gamma^t C(X_t, \pi(X_t)) \middle| X_0 = x\right], 0 < \gamma < 1, x \in X,$$

which is a performance measure of the policy π.

Given a fixed initial state probability distribution δ defined over X, we define the *average value of π for δ*:

$$J_\delta^\pi = \sum_{x \in X} V^\pi(x)\delta(x)$$

and the *average optimal value with δ*:

$$J_\delta^* = \min_{\pi \in \Pi} J_\delta^\pi.$$

The goal is to find an optimal policy $\pi \in \Pi$ that achieves J_δ^*. The discount factor is used to represent the degree of the emphasis on the immediate future performance. As the value of γ gets closer to 0, we emphasize the immediate future performance much more than the distant future performance and as the value of γ gets closer to 1, we emphasize the distance future performance equally to the immediate future.

The following result, called Bellman's optimality principle, is well-known in the control literature [20].

Theorem 2.1 *Let $V^*(x) = \min_{\pi \in \Pi} V^\pi(x), x \in X$. For all $x \in X$,*

$$V^*(x) = \min_{a \in A(x)} \left\{ C(x,a) + \gamma \sum_{y \in X} P(y|x,a)V^*(y) \right\}$$

and $V^*(x)$, $x \in X$ is unique and a policy π^* defined as

$$\pi^*(x) \in \arg\min_{a \in A(x)} \left\{ C(x,a) + \gamma \sum_{y \in X} P(y|x,a)V^*(y) \right\}, x \in X$$

satisfies that $V^{\pi^*}(x) = V^*(x), x \in X$, achieving J_δ^* for any δ.

There exist two well-known algorithms to compute π^* or the optimal value function V^*. VI starts with an arbitrarily chosen (bounded) function defined over X and updates the function into a new function defined over X. PI starts with arbitrarily chosen initial policy and generates a new policy which improves the old policy. PI computes π^* in a finite number of steps because there are a finite number of policies in Π and PI preserves the monotonicity in terms of the policy performance. The PI algorithm consists of two parts: policy evaluation and policy improvement. We provide a formal description of PI below.

Let $B(X)$ be the space of real-valued bounded measurable functions on X. We define an operator $T : B(X) \to B(X)$ as

$$T(\Phi)(x) = \min_{a \in A(x)} \left\{ C(x,a) + \gamma \sum_{y \in X} P(y|x,a)\Phi(y) \right\}, \Phi \in B(X), x \in X \tag{1}$$

and similarly, an operator $T_\pi : B(X) \to B(X)$ for $\pi \in \Pi$ as

$$T_\pi(\Phi)(x) = C(x, \pi(x)) + \gamma \sum_{y \in X} P(y|x, \pi(x))\Phi(y), \Phi \in B(X), x \in X. \tag{2}$$

It is well known (see, e.g., [20]) that for each policy $\pi \in \Pi$, there exists a corresponding unique $\Phi \in B(X)$ such that for $x \in X$,

$$T_\pi(\Phi)(x) = \Phi(x) \text{ and } \Phi(x) = V^\pi(x).$$

The policy evaluation step obtains V^π for a given π, and the policy improvement step obtains $\hat{\pi} \in \Pi$ such that

$$T(V^\pi)(x) = T_{\hat{\pi}}(V^\pi)(x), x \in X.$$

The policy $\hat{\pi}$ improves π in that $V^{\hat{\pi}}(x) \leq V^\pi(x)$ for all $x \in X$.

The running time-complexity of VI is polynomial in $|X|$, $|A|$, and $1/(1-\gamma)$ and in particular one iteration takes $O(|X|^2 \cdot |A|)$ time. For PI, doing policy improvement step takes $O(|X|^2 \cdot |A|)$ time. See [17] for a detailed discussion, including the state and action space dependent time-complexity of the linear programming approach for solving MDPs. Therefore, applying the exact methods for solving MDPs is very difficult if the state and/or the action space are large, which is true for many interesting problems. We will discuss complexity issues further after we present our algorithm in Section 5.

3 ANT-PI Algorithm

3.1 Algorithm description

A high-level description of ANT-PI is shown in Figure 1, where some steps are described at a conceptual level, with details provided in the following subsections.

Ant System Policy Iteration (ANT-PI)

- **Initialization:**

  Select the number of ants n. Set $k = 1$.

  Select $\mu \in (0, 1]$ and a positive function φ and an arbitrary $\pi_0^e \in \Pi$ and set $J_\delta^{\pi_0^e} = \infty$.

  Initialize the pheromone for each ant: set $\tau_0(x, a) = 1/|A(x)|$ for all $x \in X$ and $a \in A(x)$.

  Initialize the policy generation function: set $\rho_0(x, a) = 1/|A(x)|$ for all $x \in X$ and $a \in A(x)$.

- **Repeat:**

  - **Policy Generation (Walks of Ants):**

    * Generate $\Lambda_k = \{\pi_k^i, i = 1, ..., n\}$, where π_i^k is generated using ρ_{k-1}.
    * Obtain $J_\delta^{\pi_k^i}$ for each $\pi \in \Lambda_k$.

  - **Pheromone Update:**

    * For each $i = 1, ..., n$, for each $x \in X$ and $a \in A(x)$,

    $$\tau_k^i(x, a) = \begin{cases} \varphi(J_\delta^{\pi_k^i}) \text{ if } \pi_k^i(x) = a \\ 0 \text{ otherwise.} \end{cases} \tag{3}$$

    * For each $x \in X$ and $a \in A(x)$,

    $$\tau_k(x, a) = \begin{cases} \tau_{k-1}(x, a) \text{ if } \sum_{x' \in X, a' \in A(x')} \sum_{i=1}^n \tau_k^i(x', a') = 0 \\ (1 - \mu)\tau_{k-1}(x, a) + \mu \cdot \frac{\sum_{i=1}^n \tau_k^i(x,a)}{\sum_{x' \in X, a' \in A(x')} \sum_{i=1}^n \tau_k^i(x', a')}, \text{ otherwise.} \end{cases} \tag{4}$$

  - **Policy Generation Function Update:**

    * For each $x \in X$ and $a \in A(x)$,

    $$\rho_k(x, a) = \frac{\tau_k(x, a)}{\sum_{x' \in X, a' \in A(x')} \tau_k(x', a')}.$$

  - **Elite Policy Generation (Policy Switching):**

    * Generate the elite policy of Λ_k defined as

    $$\pi_k^e(x) \in \left\{ \underset{\pi \in \Lambda_k \cup \{\pi_{k-1}^e\}}{\arg\min} \left(V^\pi(x)\right)(x) \right\}, x \in X. \tag{5}$$

  - $k \leftarrow k + 1$

Figure 1: Ant System Policy Iteration (ANT-PI)

3.2 Policy generation

We can view the set Π of all possible stationary policies as a "construction graph" in the ant system [15]. The construction graph is a pair (G, Ω), where $G = (V, E)$ is a directed graph such that there is a unique starting vertex or node from which each ant begins traversing the graph. Each directed *walk* of an ant in G contains each node of G at most once and the last node on the walk has no successor node in G, that is not already contained in the walk. The function Ω maps the set of ants' feasible walks to the set of feasible solutions of a given optimization problem. That is, the construction graph specifies a particular encoding of the feasible solutions as walks of ants. The objective function value of the walk is set equal to the objective function value of the corresponding feasible solution of the original problem.

In our contexts, we arrange the states in X in an arbitrary order $\mathcal{O}$. Each state $x \in X$ is a node $v \in V$ in the construction graph and each arc e corresponds to a pair (x, a) of a state $x \in X$ and an admissible action $a \in A(x)$ at the state and the direction of the arc reaches from the current state x to the next state in the order $\mathcal{O}$. A particular ant traverses all the states in the construction graph following the order $\mathcal{O}$ from the first state of $\mathcal{O}$. When it moves from a state x to another state y, it traverses randomly the arc $e = (x, a)$ with the transition probability specified by $\rho_{k-1}(x, a)$, $a \in A(x)$, where k is the particular iteration step of ANT-PI and it is initialized with uniform distribution such that $\rho_0(x, a) = 1/|A(x)|$ for all $x \in X$ and $a \in A(x)$. It can be directly seen that once an ant finishes the tour, it has generated a stationary policy $\pi \in \Pi$ at random.

After random stationary policies $\pi_k^1, ..., \pi_k^n$ are generated from the walks of the ants, each policy is evaluated obtaining the value of following the policy. ANT-PI assumes that the state transition probabilities are known to the system. We discuss the case where the probabilities are unknown later. For each policy $\pi \in \Lambda_k = \{\pi_k^1, ..., \pi_k^n\}$, the following set of linear equations are solved for $V^\pi(x), x \in X$:

$$V^\pi(x) = C(x, a) + \gamma \sum_{y \in X} P(y|x, a) V^\pi(y), x \in X,$$

obtaining J_δ^π by averaging $V^\pi(x)$ over δ.

Once this "forward" phase (walking from the starting state to the ending state in $\mathcal{O}$) is finished, the next phase is "backward". Each ant traverses back along the path it travelled and deposits its pheromone on the path. Depositing its pheromone along the path is like expressing its opinion on the policy he generated to the other ants. In the next subsection, we study how this pheromone is generated.

3.3 Pheromone and policy generation function update

While each ant backtracks the path he traversed over the construction graph at iteration k, each ant $i = 1, ..., n$ modifies the pheromone τ_k^i. The pheromone is initialized with uniform distribution such that $\tau_0(x, a) = \frac{1}{|A(x)|}$ for all $x \in X$ and $a \in A(x)$. At iteration k, for each $i = 1, ..., n$, for each $x \in X$ and $a \in A(x)$,

$$\tau_k^i(x, a) = \begin{cases} \varphi(J_\delta^{\pi_k^i}) \text{ if } \pi_k^i(x) = a \\ 0 \text{ otherwise,} \end{cases} \tag{6}$$

where φ is a non-increasing function which may depend on the walks of the ants at the iterations $1, ..., k-1$. (We defer the discussion on the function φ and the condition for the use of φ to the next subsection.) We then *merge* each ant's pheromone: for each $x \in X$ and $a \in A(x)$,

$$\tau_k(x, a) = \begin{cases} \tau_{k-1}(x, a) \text{ if } \sum_{x' \in X, a' \in A(x')} \sum_{i=1}^n \tau_k^i(x', a') = 0 \\ (1-\mu)\tau_{k-1}(x, a) + \mu \cdot \frac{\sum_{i=1}^n \tau_k^i(x,a)}{\sum_{x' \in X, a' \in A(x')} \sum_{i=1}^n \tau_k^i(x', a')}, \text{ otherwise.} \end{cases} \tag{7}$$

The constant $\mu \in (0, 1]$ is called the *evaporation factor* (see, e.g., [11]). If no walk is rewarded ($\sum_{x' \in X, a' \in A(x')} \sum_{i=1}^n \tau_k^i(x', a') = 0$), the pheromones for all pairs of $(x, a), x \in X, a \in A(x)$ do not change. Otherwise, only an amount of $1 - \mu$ of pheromone remains on the graph after iteration k. The remaining amount of μ is the "budget" for rewarding the walks traversed at iteration k according to their respective average value of following their (randomly generated) policies. Each ant i can be thought as depositing the amount of $\tau_k^i(x, a)$ to express the degree of "goodness" on its walk at iteration k. Then, proportionally to the reported values $\tau_k^i(x, a)$, the actual pheromone is updated within the budget of μ.

By the update rule, the pheromone on "promising" arcs of the construction graph is increased, such that those arcs are traversed more often by the ants in the future. At the extreme case of $\mu = 0$, the system ignores the impact of the value function on the walks of the ants, which is excluded in our setting.

By the merged pheromone value, the policy generation function is updated yielding a new probability distribution over the arcs: for each $x \in X$ and $a \in A(x)$,

$$\rho_k(x, a) = \frac{\tau_k(x, a)}{\sum_{x' \in X, a' \in A(x')} \tau_k(x', a')}.$$

For improving the performance of the algorithm, *visibility* values $\eta_k(x, a)$ can be introduced into the probability computation: Arcs that seem promising in some *a priori* consideration can be favored. If this option is chosen, one has to replace the last formula above by

$$\rho_k(x,a) = \frac{\tau_k(x,a)\cdot\eta_k(x,a)}{\sum_{x'\in X, a'\in A(x')}\tau_k(x',a')\cdot\eta_k(x',a')},$$

where $\eta_k(x,a) > 0$ is a pre-evaluation value of action a in state x. The simplest way to define $\eta_k(x,a)$ is

$$\eta_k(x,a) = \frac{1}{[C(x,a)+c]^\alpha}$$

for each k, with real numbers $c > 0$, $\alpha > 0$. It is advisable to choose α rather small (say: $\alpha = 0.1$ to 0.5) in order not to bias the learning process governed by the pheromone values $\tau_k(x,a)$ *too* much towards the "greedy" pre-evaluations $\eta_k(x,a)$. Note that the values $\eta_k(x,a)$ as defined above are always positive.

3.4 Elite policy generation

The function φ determines how the objective function value of a walk of an ant (i.e., the average value of following a policy) is rewarded into its pheromone. The convergence of Gutjahr's ant system is based on *elitist strategy* [11] [12] similar to the "*elitism*" concepts introduced by De Jong [8] for GA. In this elitist strategy, the walks that are dominated by another walk that was already traversed do not get pheromone increments anymore. Only the best walks are rewarded.

From the update rule of the pheromone in Figure 1, in order to guarantee the convergence of the algorithm, one has to choose φ such that φ satisfies that for $\pi \in \Lambda_k, k = 1, 2, ...,$

$$\begin{aligned} \varphi(J^\pi_\delta) &> 0 \text{ for } J^\pi_\delta \le J^{\pi^e_{k-1}}_\delta \\ \varphi(J^\pi_\delta) &= 0 \text{ otherwise.} \end{aligned}$$

We can see that only walks that are at least as good as the performance of the policy π^e_{k-1} get a positive increment for the pheromone. If π^e_{k-1} is equal to the policy that achieves

$$\min_{\pi\in\cup_{i=1}^{k-1}\Lambda_i} J^\pi_\delta,$$

then we have the elitist strategy used in Gutjahr's ant system. In particular, if a cost function C is positive, we can set $\varphi(J^\pi_\delta) = 1/J^\pi_\delta$.

It turns out that we can extend the elitist strategy further. The recent work of policy switching by Chang et al. [6] provides a formal way of combining a finite number of policies in a set to generate an improved policy of all the policies in the set, which we call *elite* policy for the set. It is in general difficult to have a formal rule like policy switching for plain combinatorial optimization problems. Via policy switching, we can change the controlling condition of the elitist strategy so that a walk generated at kth iteration gets rewarded in the pheromone update only if the objective function value of the walk is no worse than the performance of the elite policy of the set $\cup_{i=1}^{k-1}\Lambda_i$. This would make a policy or a walk get rewarded only if the quality of the policy is no worse than any policy generated up to now by ants. By recursively defining the elite policy of kth iteration over the set $\Lambda_k \cup \{\pi_{k-1}^e\}$, the controlling condition of the pheromone update can also include all of the elite policies generated up to now, so that any policy generated by ants so far is included.

Given a nonempty subset Δ of Π, we define a policy $\bar{\pi}$ generated by policy switching with respect to Δ as

$$\bar{\pi}(x) \in \{\arg\min_{\pi \in \Delta}(V^{\pi}(x))(x)\}, x \in X. \tag{8}$$

We now show that the policy generated by policy switching improves any policy in Δ for the completeness (see Theorem 3 in [6].)

Theorem 3.1 *Consider a nonempty subset Δ of Π and the policy $\bar{\pi}$ generated by policy switching with respect to Δ given in Equation (8). Then, for all $x \in X$,*

$$V^{\bar{\pi}}(x) \leq \min_{\pi \in \Delta} V^{\pi}(x).$$

Proof: We begin with a lemma, which states a basic property of the T_π-operator.

Lemma 3.1 *Given $\pi \in \Pi$, suppose there exists $\Phi \in B(X)$ for which*

$$T_\pi(\Phi)(x) \leq \Phi(x), x \in X. \tag{9}$$

Then, $V^{\pi}(x) \leq \Phi(x)$ for all $x \in X$.

Proof: By successive applications of the T_π-operator to both sides of Equation (9) and the monotonicity property of the operator, we have that for all $x \in X$,

$$\lim_{n\to\infty} T_\pi^n(\Phi)(x) \leq \Phi(x).$$

It is well-known that T_π is a contraction mapping in $B(X)$ and that iterative application of T_π on any initial value function converges monotonically to the fixed point V^{π}. Therefore, $\lim_{n\to\infty} T_\pi^n(\Phi)(x) = V^{\pi}(x), x \in X$, which proves the lemma. ∎

Now define $\Phi(x) = \min_{\pi \in \Delta} V^\pi(x)$ for all $x \in X$. Pick an arbitrary state $x \in X$. From the definition, there exists a policy $\pi' \in \Delta$ such that $V^{\pi'}(x) \leq V^\pi(x)$ for all $\pi \in \Delta$ and $\bar{\pi}(x) = \pi'(x)$. It follows that

$$\begin{aligned} T_{\bar{\pi}}(\Phi)(x) &= R(x, \bar{\pi}(x)) + \gamma \sum_{y \in X} P(x, \bar{\pi}(x))(y)\Phi(y) \\ &= R(x, \pi'(x)) + \gamma \sum_{y \in X} P(x, \pi'(x))(y)\Phi(y) \\ &\leq R(x, \pi'(x)) + \gamma \sum_{y \in X} P(x, \pi'(x))(y)V^{\pi'}(y) \\ &= V^{\pi'}(x) = \Phi(x). \end{aligned}$$

By the lemma above, the claim is proved. ∎

The above theorem immediately implies the following result.

Corollary 3.1 *Consider a nonempty subset Δ of Π and the policy $\bar{\pi}$ generated by policy switching with respect to Δ given in Equation (8). Then, for any initial state distribution δ,*

$$J_\delta^{\bar{\pi}} \leq \min_{\pi \in \Delta} J_\delta^\pi.$$

Proof: By the definition of $\bar{\pi}$ (see Equation (8)), for any fixed $\pi \in \Delta$ and $x \in X$,

$$V^{\bar{\pi}}(x) \leq V^\pi(x).$$

Therefore,

$$J_\delta^{\bar{\pi}} = \sum_{x \in X} V^{\bar{\pi}}(x)\delta(x) \leq \sum_{x \in X} V^\pi(x)\delta(x) = J_\delta^\pi,$$

which implies that

$$J_\delta^{\bar{\pi}} \leq \min_{\pi \in \Delta} J_\delta^\pi.$$

∎

We generate a policy π_k^e, the elite policy with respect to the set $\Lambda_k \cup \{\pi_{k-1}^e\}$, which improves any policy in Λ_k and π_{k-1}^e via policy switching. Note that this is different from the elitist concept of De Jong [8], where the elitist is the best policy in a "population" of $\Lambda_k \cup \{\pi_{k-1}^e\}$. That is, the elitist is a policy that achieves $\min_{\pi \in \Lambda_k \cup \{\pi_{k-1}^e\}} J_\delta^\pi$. Therefore, this immediately implies that the following monotonicity holds.

Corollary 3.2 *For any δ and for all $k \geq 0$, any δ,*

$$J_\delta^{\pi_k^e} \leq J_\delta^{\pi_{k-1}^e}.$$

The policy switching step is one of the key parts in ANT-PI to *speed up* the convergence of ANT-PI. Consider a set S of two policies π_1 and π_2 and let $\Psi = \{x | \pi_1(x) \neq \pi_2(x), x \in X\}$. Write $\pi < \pi'$ if for all $x \in X$,

$$V^\pi(x) \leq V^{\pi'}(x)$$

and for some state $x \in X$,

$$V^\pi(x) < V^{\pi'}(x)$$

and write $\pi \leq \pi'$ if for all $x \in X$, $V^\pi(x) \leq V^{\pi'}(x)$. Then there are at least $|\Psi|$ policies $\hat{\pi}_j$, $j = 1, ..., |\Psi|$ such that for each j, either

$$\pi(S) \geq \hat{\pi}_j > \pi_1 \text{ or } \pi(S) \geq \hat{\pi}_j > \pi_2$$

holds, where $\pi(S)$ is the policy generated from policy switching on S. In other words, by one application of policy switching, we eliminate at least $|\Psi|$ policies but at most $|X|$ policies in the search process. This is because given a policy π, if we can improve the policy π by modifying the actions in m states, we rule out at least m policies that are better than π. See Lemma 5 [18] for a formal proof.

As we can see, policy switching directly manipulates policies to generate an improved policy relative to all policies it was applied to, eliminating the operation of minimization over the entire action space, which is the main computational advantage that replaces the policy improvement step in the original PI.

3.5 Convergence

Even if the average value of following the two consecutive elite policies are identical, this does not necessarily mean that the elite policy is an optimal policy as in PI. Still, we have a probabilistic convergence guarantee for ANT-PI.

In [15], the following convergence result has been proved. We cite it here within the context of the special situation of the present paper.

Theorem 3.2. *Assume that the following conditions are satisfied:*

(1) There is only one optimal walk through the construction graph.

(2) Along the optimal walk, the visibility values satisfy $\eta_k(x,a) > 0$.

(3) Let $J^*(k)$ be the lowest cost value observed in the iterations $1, \ldots, k-1$, i.e., the lowest objective function value $J_\delta^{\pi_k^i}$ corresponding to a walk of an ant in these $k-1$ iterations. (In the case of iteration $k = 1$, let $J^*(k) = \infty$.) Then the reward function $\varphi = \varphi_k$ satisfies

(i) $\varphi_k(J_\delta^{\pi_k^i}) > 0$ for $J_\delta^{\pi_k^i} \leq J^*(k)$,

(ii) $\varphi_k(J_\delta^{\pi_k^i}) = 0$ for $J_\delta^{\pi_k^i} > J^*(k)$

for each k.

Moreover, let P_m denote the probability that a fixed ant traverses the optimal walk in iteration m. Then the following two assertions are valid:

1. *For each $\epsilon > 0$ and for fixed parameter μ, it can be achieved by the choice of a sufficiently large number n of ants that $P_m \geq 1 - \epsilon$ holds for all $m \geq m_0$ (with an integer m_0 depending on ϵ).*

2. *For each $\epsilon > 0$ and for fixed parameter n, it can be achieved by the choice of an evaporation factor μ sufficiently close to zero that $P_m \geq 1 - \epsilon$ holds for all $m \geq m_0$ (with an integer m_0 depending on ϵ).*

In [15], this result is proved with the help of four lemmas. Here, we need a slightly stronger version of this theorem:

Theorem 3.3. *The assertion of Theorem 3.2 is also valid if condition (i) is replaced by the weaker condition*

(i') $\varphi_k(J_\delta^*) > 0$ for the cost value J_δ^* of the optimal solution.

Proof: The proof is a repetition of the proof of Theorem 4.1 in [15], with very slight modifications: First of all, Lemma 4.1 in [15] and its Corollary (which generalizes the lemma) remain unchanged. The Corollary states the following:

The conditional probability that at least one ant traverses in iteration m the optimal walk, given that no ant has traversed the optimal walk in one of the previous iterations, is larger or equal to $1-(1-c^{m-1}p)^n$, where $c>0$, $p \in]0,1[$, and n is the number of ants.

Condition (3) is not used for proving this statement, so the replacement of condition (i) by (i') does not enter into the proof.

Lemmas 4.2 to 4.4 in [15] make assertions on conditional probabilities, conditioned on the event that in iteration m, the optimal walk is traversed for the first time, and refer only to iterations k with $k>m$. By condition (i') and since $J_\delta^{\pi_k^i}$ cannot be smaller than J_δ^*,

$$\varphi_k(J_\delta^{\pi_k^i}) > 0 \text{ if } J_\delta^{\pi_k^i} \leq J_\delta^*.$$

Furthermore, in iterations $k>m$, the equation $J^*(k) = J_\delta^*$ must hold, since the optimal solution has already been observed. Therefore, in all cases where condition (i) is required in the proof, it is satisfied indeed. We conclude that also Lemmas 4.2 to 4.4 of [15] hold.

Finally, the proof of Theorem 4.1 in [15] is based only on the Corollary to Lemma 4.1 and on Lemma 4.4, so it does not use condition (i) directly. ■

We are now able to use this general result to obtain a convergence result on ANT-PI:

Theorem 3.4. *For the algorithm ANT-PI, the following is true: Assume that there is only one optimal policy and that all chosen visibility values are strictly positive, and let P_m denote the probability that ANT-PI produces the optimal policy in iteration m. Then the following two assertions are valid:*

1. *For each $\epsilon>0$ and for fixed parameter μ, it can be achieved by the choice of a sufficiently large number n of ants that $P_m \geq 1-\epsilon$ holds for all $m \geq m_0$ (with an integer m_0 depending on ϵ).*

2. *For each $\epsilon>0$ and for fixed parameter n, it can be achieved by the choice of an evaporation factor μ sufficiently close to zero that $P_m \geq 1-\epsilon$ holds for all $m \geq m_0$ (with an integer m_0 depending on ϵ).*

Proof: The result follows directly from Theorem 3.3. Condition (1), as stated in Theorem 3.2, is satisfied, since we have encoded the policies in such a way that each policy is represented by only one walk (and vice versa), and the optimal *policy* is unique by assumption. Condition (2) is also satisfied by assumption. That condition (3) with (i) replaced by (i') holds can be seen as follows:

First, observe that according to the special choice of φ at the beginning of Subsection 3.4, $\varphi(J^{\pi}_{\delta}) > 0$ for $J^{\pi}_{\delta} \leq J^{\pi^e_{k-1}}_{\delta}$. Now, since J^*_{δ} is the minimum cost value, $J^*_{\delta} \leq J^{\pi^e_{k-1}}_{\delta}$, which implies $\varphi(J^*_{\delta}) > 0$, i.e., condition (i') holds.

Secondly, again according to the special choice of φ at the beginning of Subsection 3.4, $\varphi(J^{\pi}_{\delta}) = 0$ for $J^{\pi}_{\delta} > J^{\pi^e_{k-1}}_{\delta}$. By Theorem 3.1, the policy $\pi^e_{k-1} \in \Lambda_{k-1} \cup \{\pi^e_{k-2}\}$ obtained by policy switching satisfies $J^{\pi^e_{k-1}}_{\delta} \leq J^*(k)$. Therefore, $\varphi(J^{\pi}_{\delta}) = 0$ for $J^{\pi}_{\delta} > J^*(k)$, which is condition (ii).

In total all conditions (1) – (3) of Theorem 3.4 are fulfilled, so the indicated convergence property holds. ∎

Note that the condition of the uniqueness of the optimal policy can always be achieved by a very slight change of the cost function that does not make any suboptimal policy appear as optimal: Let $\chi(\pi)$ denote the index of policy π in the chosen fixed order. Add $\epsilon \cdot \chi(\pi)$ to the cost value of each policy π, where ϵ is small enough. Then uniqueness of the optimal solution is obtained.

4 ANT-TD: Combining with temporal difference learning

It has been so far assumed that the state transition dynamics P of a given MDP is known. When P is unknown to the ant system, we can use one of the well-known reinforcement learning algorithms called *temporal difference* learning (TD-learning) for evaluating the average value of following a given policy. We briefly study TD(λ)-learning. For a substantial discussion, see, e.g., [21].

The main idea of TD-learning is based on using the difference between temporally successive predictions given at every time step as a heuristic error signal for learning (estimating the average value of following a given policy). Given that a prediction error has been observed at a given time step, we emphasize more recent error feedbacks in time with an exponentially decaying parameter λ, where this derives previous estimates for previously (visited) states are also corrected.

Formally, given with a parameter $\lambda \in [0, 1]$, an agent starts from an initial state x_0. At time t, the agent takes an action $a = \pi(x_t)$ at state x_t and reaches to the next state x_{t+1}. He observes the cost $C(x_t, a)$ of taking a at x_t and updates the estimated value function $\hat{V}_t$ by

$$\forall x \in X, \hat{V}_{t+1}(x) \leftarrow \hat{V}_t(x) + \alpha_t(x) z_t(x) d_t,$$

where $\hat{V}_t$ is the current estimate of the value function V^π at time t. $\alpha_t(x)$ is the stepsize and $z_t(x)$ is the "eligibility" trace vector of size $|X|$, which is given by the following iterative equation: for all $x \in X$,

$$z_t(x) = \begin{cases} \gamma\lambda z_{t-1}(x) \text{ if } x \neq x_t \\ \gamma\lambda z_{t-1}(x) + 1 \text{ if } x = x_t. \end{cases} \tag{10}$$

We can see that the eligibility traces decay exponentially with $\gamma\lambda < 1$, that is, recency is weighted more than latency, and when a state is visited, its trace is increased by 1. Finally, d_t is the temporal difference error given by

$$d_t = R(x_t, \pi(x_t)) + \gamma\hat{V}_t(x_{t+1}) - \hat{V}_t(x_t).$$

It has been shown that this algorithm converges with probability 1 to V^π under some general conditions on the stepsizes and on the Markov chain induced by π [3]. For efficient implementation of this algorithm, see, e.g., [7] [13].

We apply TD(λ)-learning for a finite number of time steps to evaluate the performance of a randomly generated policy by each ant. From this, we update the pheromone and the policy generation function and the elite policy, having a heuristic PI approach (ANT-TD) combined with a reinforcement learning for solving MDPs.

We finally remark that Gambardella and Dorigo [12] devised an ant-system based algorithm that combines with a reinforcement learning called Q-learning (see, e.g., [21]), specifically to solve a travelling salesman problems, not considering stochastic optimization problem. Our approach is fundamentally different from theirs.

5 Complexity issue

It is known that PI converges faster to the optimal value than VI if both algorithms begin with the same value [20] and that VI is usually outperformed by PI in practical applications. For this reason, we discuss here only complexity issues between ANT-PI and PI.

ANT-PI replaces the single policy evaluation step of PI with multiple policy evaluations, where each policy is generated randomly by an ant, and replaces the single policy improvement step of PI with a combination of a standard ant system framework with policy switching. Doing the single policy evaluation requires solving a system of $|X|$ equations, which takes $O(|X|^3)$ time by

Gaussian elimination and $O(|X|^{2.8})$ by the fastest known algorithms. ANT-PI does not add any overhead for this step compared with PI because multiple policy evaluations can be done in parallel independently. At the worst case, it increases the time complexity of policy evaluation by the factor of the number of ants.

As we mentioned before, the policy improvement step takes $O(|X|^2 \cdot |A|)$ time. If $|A|$ is large, e.g., $O(|X|^2)$, policy improvement takes too much time. If we use an approximation to alleviate this problem, that is, replacing minimization by, e.g., a randomized search, we cannot guarantee the convergence to an optimal policy. On the other hand, policy switching takes $O(n|X|)$ time by directly manipulating policies, where n is the number of ants. Therefore, the main advantage of ANT-PI over PI is for the problems whose action space is large.

The other factor for complexity is the number of iterations required. To the authors' best knowledge, there is no known formal result on this issue for PI except that a finite number of iterations for the convergence is sufficient (under our finite state and action spaces model). PI may need a very large number of iterations, whereas by the ant system, a good approximative solution may already be reached after a comparatively smaller number of iterations. Note also that the sequence of generated elite policies has a monotonicity property as in the sequence of generated policies in PI. Therefore, if both PI and ANT-PI start with a "good" policy and if PI converges to an optimal policy relatively fast, we expect that ANT-PI also converges to an optimal policy relatively fast.

Suppose that the action space A is not that large, making the direct manipulation of policies by policy switching not helpful. Then, we can replace **Elite Policy Generation** step with a method called "parallel rollout" [6], which is a generalization of PI with *multiple policy* improvement. We simply replace Equation (5) in ANT-PI with

$$\pi_k^e(x) \in \arg\min_{a \in A(x)} \left\{ C(x,a) + \gamma \sum_{y \in X} P(y|x,a) \min_{\pi \in \Lambda_k \cup \{\pi_{k-1}^e\}} V^\pi(y) \right\}.$$

Then, every property we discussed before still holds and it is guaranteed that the resulting ANT-PI is faster than PI in terms of the number of iterations (with the same initialization for both algorithms). Also in sequential implementations, it increases the time complexity of policy evaluation only by the factor of the number of ants.

6 Concluding Remarks

ANT-PI is naturally parallelizable. By doing so, we can improve the runtime. Basically, we partition the policy space Π into subsets of $\{\Pi_i\}$ such that $\bigcup_i \Pi_i = \Pi$ and $\Pi_i \cap \Pi_j = \emptyset$ for all $i \neq j$. Note that there is a corresponding construction graph to each Π_i. We then apply ANT-PI into each Π_i in parallel and then once each part terminates, the best policy π_i^* from each part is taken. We apply then policy switching to the set of best policies $\{\pi_i^*\}$. For a general result regarding parallelization of an algorithm in this respect, see [5].

To obtain a "good" probability bound on the performance of the ant system algorithm in [15], the number of ants and/or evaporation factor needs to be large for applying the algorithm in practice. However, as noted in [15], the analysis is based on a very coarse estimation of the probability that an ant finds the optimal solution so that convergence to the optimum with high probability may be achieved with a moderate number of agents. Following the argument in [15], the purpose of this paper is to demonstrate the capability of the ant-system based approach, in the theoretical perspective, for solving dynamic stochastic optimization problems. Certainly, the future works must be implementing and evaluating the proposed algorithms for difficult MDP problems.

References

[1] E. Altman, "Applications of Markov decision processes in communication networks: a survey," *Markov Decision Processes, Models, Methods, Directions, and Open Problems*, E. Feinberg and A. Shwartz (Eds.) Kluwer, pp. 488–536, 2001.

[2] D. P. Bertsekas, *Dynamic Programming and Optimal Control, Volumes 1 and 2.* Athena Scientific, 1995.

[3] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro Dynamic Programming.* Athena Scientific, 1996.

[4] H. S. Chang, M. Fu, and S. I. Marcus, "An asymptotically efficient algorithm for finite horizon stochastic dynamic programming problems," in *Proc. of the 42nd IEEE Conf. Decision and Control*, 2003, to appear.

[5] H. S. Chang, H-G. Lee, M. Fu, and S. I. Marcus, "Evolutionary policy iteration for solving Markov decision processes," *IEEE Trans. on Automatic Control*, revised, 2003.

[6] H. S. Chang, R. Givan, and E. K. P. Chong, "Parallel rollout for on-line solution of partially observable Markov decision processes," *Discrete Event Dynamic Systems: Theory and Application*, revised, 2003.

[7] P. Cichosz, "Truncating temporal differences: on the efficient implementation of TD(λ) for reinforcement learning," *Journal of Artificial Intelligence Research*, vol. 2, pp. 287–318, 1995.

[8] K. A. De Jong, *An Analysis of the Behavior of a Class of Genetic Adaptive Systems,* Ph.D. Thesis, Univ. of Michigan, Ann Arbor, MI, 1975.

[9] M. Dorigo and G. Di Caro, "The ant colony optimization metaheuristic," *New Ideas in Optimization*, D. Corne, M. Dorigo (eds.), pp. 11-32, McGraw-Hill, NY, USA, 1999.

[10] M. Dorigo, G. Di Caro, and T. Stützle, Special Issue on "Ant Algorithms," *Future Generation Computer Systems*, vol. 16, no. 8, 2000.

[11] M. Dorigo, V. Maniezzo, A. Colorni, "The Ant System: optimization by a colony of cooperating agents," *IEEE Trans. Systems Man Cybernet.,* vol. 25, pp. 29–41, 1996.

[12] L. M. Gambardella and M. Dorigo, "Ant-Q: a reinforcement learning approach to the traveling salesman problem," *Proc. of the 12th Int. Conf. on Machine Learning*, 1995, pp. 252–260.

[13] F. Garcia and F. Serre, "Efficient asymptotic approximation in temporal difference learning," *Proc. of the 14th European Conf. on Artificial Intelligence*, 2000, pp. 296–300.

[14] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning,*, Addison-Wesley, 1989.

[15] W. J. Gutjahr, "A graph-based ant system and its convergence," *Future Generation Computer Systems*, vol. 16, pp. 873–888, 2000.

[16] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, pp. 45–54, 1983.

[17] M. Littman, T. Dean, and L. Kaelbling, "On the complexity of solving Markov decision problems," in *Proc. 11th Annual Conf. on Uncertainty in Artificial Intelligence*, 1995, pp. 394–402.

[18] Y. Mansour and S. Singh, "On the complexity of policy iteration," in *Proc. UAI*, 1999, pp. 401–408.

[19] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, NJ, 1982.

[20] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* Wiley, New York, 1994.

[21] R. Sutton and A. Barto, *Reinforcement Learning.* MIT Press, 2000.